I hereby certify that this correspondence is being filed via
EFS-Web with the United States Patent and Trademark Office
on October 25, 2007.

TOWNSEND and TOWNSEND and CREW LLP

By: _____ /Janet L. Newmaker/ _____
                Janet L. Newmaker

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	Confirmation No.: 5815
Geoffrey R. Hird	Examiner: Paula W. Klimach
Application No.: 09/874,795	Art Unit: 2135
Filed: June 5, 2001	
For: Method And Apparatus For Secure Cryptographic Key Generation, Certification And Use	**COMMUNICATION AFTER NOTICE OF ALLOWANCE**
Customer No.: 20350	

Mail Stop Issue Fee
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In response to the Notice of Allowance mailed September 6, 2007, a review of the file was performed. An Information Disclosure Statement (IDS) filed August 15, 2002, does not appear to have been fully considered. The non-patent literature submitted with that IDS was not initialed by the Examiner. This IDS was timely filed in accordance with 37 C.F.R. § 1.97(b)(3). Copies of the partially initialed SB/08, the IDS, the non-patent literature, and the PTO-stamped postcard acknowledging receipt are enclosed for the Examiner's convenience.

The Applicant requests that the Examiner review the non-patent literature and return a fully initialed SB/08 to the undersigned attorney.

Respectfully submitted,

Date: <u>October 25, 2007</u>

/Irvin E. Branch/
Irvin E. Branch
Reg. No. 42,358

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, Eighth Floor
San Francisco, CA 94111-3834
Tel: 303-571-4000
Fax: 415-576-0300
IEB/jln
61191516 v1

INFORMATION DISCLOSURE CITATION
(Use several sheets if necessary)

Docket Number (Optional)	Application Number
028410-0016	09/874,795

Applicant(s)	
Geoffrey R. Hird	AUG 1 9 2002

Filing Date	Group Art Unit
June 5, 2001	2131

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL	REF	DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
PWK		3,798,605	3/19/74	Feistel	340	172.5	6/30/71
PWK		5,206,905	4/27/93	Lee et al.	380	4	11/19/90
PWK		5,668,876	9/16/97	Falk et al.	380	25	6/24/94
PWK		5,894,519	4/13/99	Clemot et al.	380	23	4/9/97
PWK		6,170,058 B1	1/2/01	Kausik	713	913	12/23/97
PWK		6,408,388 B1	6/18/02	Fischer	713	176	7/30/99

RECEIVED
AUG 2 2 2002
Technology Center 2100

FOREIGN PATENT DOCUMENTS

	REF	DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	Translation YES	NO

OTHER DOCUMENTS *(Including Author, Title, Date, Pertinent Pages, Etc.)*

		Hoover, D.N. et al., "Software Smart Cards via Cryptographic Camouflage," Proceedings of the 1999 IEEE Symposium on Security and Privacy, pp. 208-215.

EXAMINER *Paula Klimach*	DATE CONSIDERED 2/17/05

EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP Section 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

TRANSMITTAL OF INFORMATION DISCLOSURE STATEMENT (Under 37 CFR 1.97(b) or 1.97(c))	Docket No. 028410-0016

In Re Application Of: **Geoffrey R. Hird**

Serial No. 09/874,795	Filing Date June 5, 2001	Examiner not assigned	Group Art Unit 2131

Title: **Method and Apparatus for Secure Cryptographic Key Generation, Certification and Use**

37 CFR 1.97(b)

1. ☒ The Information Disclosure Statement submitted herewith is being filed within three months of the filing of a national application other than a continued prosecution application under 37 CFR 1.53(d); within three months of the date of entry of the national stage as set forth in 37 CFR 1.491 in an international application; before the mailing of a first Office Action on the merits, or before the mailing of a first Office Action after the filing of a request for continued examination under 37 CFR 1.114.

37 CFR 1.97(c)

2. ☐ The Information Disclosure Statement submitted herewith is being filed after the period specified in 37 CFR 1.97(b), provided that the Information Disclosure Statement is filed before the mailing date of a Final Action under 37 CFR 1.113, a Notice of Allowance under 37 CFR 1.311, or an Action that otherwise closes prosecution in the application, and is accompanied by one of:

   ☐ the statement specified in 37 CFR 1.97(e);

   OR

   ☐ the fee set forth in 37 CFR 1.17(p).

TRANSMITTAL OF INFORMATION DISCLOSURE STATEMENT (Under 37 CFR 1.97(b) or 1.97(c))	Docket No. 028410-0016

In Re Application: **Geoffrey R. Hird**

Serial No. 09/874,795	Filing Date June 5, 2001	Examiner not assigned	Group Art Unit 2131

Method and Apparatus for Secure Cryptographic Key Generation, Certification and Use

Payment of Fee

(Only complete if Applicant elects to pay the fee set forth in 37 CFR 1.17(p))

☐ A check in the amount of   is attached.

☒ The Assistant Commissioner is hereby authorized to charge and credit Deposit Account No. **19-2385**
    as described below. A duplicate copy of this sheet is enclosed.

    ☐ Charge the amount of

    ☐ Credit any overpayment.

    ☒ Charge any additional fee required.

Certificate of Transmission by Facsimile*

I certify that this document and authorization to charge deposit account is being facsimile transmitted to the United States Patent and Trademark Office (F

_____
(Date)

Signature

Typed or Printed Name of Person Signing Certificate

Certificate of Mailing by First Class Mail

I certify that this document and fee is being deposited Aug. 15, 2002   with the U.S. Postal Service as first class mail under 37 C.F.R. 1.8 and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Signature of Person Mailing Correspondence

Jan Steele

Typed or Printed Name of Person Mailing Certificate

*This certificate may only be used if paying by deposit account.

Signature

Joseph Yang

Registration No. 41,387

Skadden, Arps, Slate, Meagher & Flom LLP

525 University Avenue

Palo Alto, California 94301

Telephone: (650) 470-4500

Dated: **August 15, 2002**

CC:

INFORMATION DISCLOSURE CITATION
(Use several sheets if necessary)

		Docket Number (Optional) 028410-001	Application Number 09/874,795
		Applicant(s) Geoffrey R. Hird	
		Filing Date **June 5, 2001**	Group Art Unit 2131

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL	REF	DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
		3,798,605	3/19/74	Feistel	340	172.5	6/30/71
		5,206,905	4/27/93	Lee et al.	380	4	11/19/90
		5,668,876	9/16/97	Falk et al.	380	25	6/24/94
		5,894,519	4/13/99	Clemot et al.	380	23	4/9/97
		6,170,058 B1	1/2/01	Kausik	713	913	12/23/97
		6,408,388 B1	6/18/02	Fischer	713	176	7/30/99

FOREIGN PATENT DOCUMENTS

	REF	DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	Translation	
							YES	NO

OTHER DOCUMENTS *(Including Author, Title, Date, Pertinent Pages, Etc.)*

		Hoover, D.N. et al., "Software Smart Cards via Cryptographic Camouflage," Proceedings of the 1999 IEEE Symposium on Security and Privacy, pp. 208-215.

EXAMINER	DATE CONSIDERED

EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP Section 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

◆IEEE

Membership   Publications/Services   Standards   Conferences   Careers/Jobs

IEEE *Xplore*
RELEASE 1.4

Welcome
United States Patent and Trademark Office

Help   FAQ   Terms   IEEE   Quick Links   ▼   » Search Result
Peer Review

Welcome to IEEE *Xplore*

- ○ Home
- ○ What Can I Access?
- ○ Log-out

Tables of Contents

- ○ Journals & Magazines
- ○ Conference Proceedings
- ○ Standards

Search

- ○ By Author
- ○ Basic
- ○ Advanced

Member Services

- ○ Join IEEE
- ○ Establish IEEE Web Account

🖨 Print Format

Your search matched **1** of **775001** documents.
Results are shown **15** to a page, sorted by **publication year** in **descending** order.
You may refine your search by editing the current search expression or entering a new one the text box.

Then click **Search Again**.

(password)and (camouflage)

Search Again

Results:
Journal or Magazine = **JNL**   Conference = **CNF**   Standard = **STD**

1 **Software smart cards via cryptographic camouflage**
Hoover, D.N.; Kausik, B.N.
Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on , 1999
Page(s): 208 -215

[Abstract]   [PDF Full-Text (72 KB)] **CNF**

Home | Log-out | Journals | Conference Proceedings | Standards | Search by Author | Basic Search | Advanced Search
Join IEEE | Web Account | New this week | OPAC Linking Information | Your Feedback | Technical Support | Email Alerting
No Robots Please | Release Notes | IEEE Online Publications | Help | FAQ | Terms | Back to Top

Software Smart Cards via Cryptographic Camouflage

D. N. Hoover and B. N. Kausik
Arcot Systems, Inc., www.arcot.com
doug@arcot.com, nat@arcot.com

Abstract

A sensitive point in public key cryptography is how to protect the private key. We outline a method of protecting private keys using cryptographic camouflage. Specifically, we do not encrypt the private key with a password that is too long for exhaustive attack. Instead, we encrypt it so that only one password will decrypt it correctly, but many passwords will decrypt it to produce a key that looks valid enough to fool an attacker. For certain applications, this method protects a private key against dictionary attack, as a smart card does, but entirely in software.

1 Introduction

With the advent of support for public key cryptography in web browsers, the use of public key cryptographic signatures and authentication protocols is becoming more common. The security of the private key, however, remains a problem. The most basic threat is the theft of a private key that is stored on a disk. Usually such a key is stored in a software key container – a file wherein the keys are encrypted by a password. An attacker who steals the container can try to guess the password using a dictionary attack. The main defenses against such an attack are (a) make the password space extremely large, (b) slow down the operation of checking a single password. Neither of these methods is very practical.

We propose a method for secure storage of private keys in software, using *cryptographic camouflage*, whereby attacks on the key container are inherently supervised. Metaphorically speaking, the proposed key container embeds the user's private key among spurious but plausible private keys. An attacker who tries to crack the key container will recover many plausible private keys, but will not be able to distinguish the correct private key from the spurious decoys until he tries to use the keys to access resources via an authentication server. The authentication server will notice multiple authentication failures and suspend access.

The main idea behind cryptographic camouflage is as follows. Usually, data that we encrypt is made up of *verifiable plaintext*, in that it has sufficient structure that will show a cryptanalyst whether he used the right key to decrypt it. For example, if an English sentence is DES-encrypted with one key and then decrypted with a different

key, the result will likely be a random-looking string of bytes, not an English sentence. Thus, an attacker can crack the encrypted message by exhaustively searching the key space, even if he does not know any of the original text— he knows he has broken the cryptogram when he recovers an English sentence. In this case, to defend against an exhaustive search attack, one must make the space of decryption keys large enough that exhaustive search attacks are computationally infeasible.

On the other hand, if every decrypt has the same structure as the original data, the attacker cannot distinguish the correctly decrypted data from the many plausible but spurious decrypts. In such case, we do not have to make our key space too large to search, because the data is *camouflaged*[1] by the many false decrypts.

What makes most conventional software key containers vulnerable to dictionary search is that they contain verifiable plaintext that shows an attacker when he has guessed the right password. The key container may contain a hash of the password or the key may be represented in a format that allows the attacker to tell if he has correctly decrypted the private key.

A more secure alternative is to store private keys on a hardware smart card. The key is be protected by a password and the smart card resists key search attacks by locking up permanently after a limited number of attempts to unlock it with incorrect passwords. Unfortunately smart cards require expensive infrastructure in the form of smart card readers, as well as substantial administrative effort in distributing and updating user cards.

Protecting a private key using cryptographic camouflage gives some of the advantages of smart cards without requiring any extra hardware. We can use camouflage to protect private keys because in most public key encryption algorithms, the part of a private key that really needs to be

---

[1] Lomas, Gong et al. [5] introduced the notion of verifiable plaintext and the more general idea of *verifiable text* [2]. They emphasize that it is unsafe to protect a secret with a key from a relatively small keyspace such as a password in the following cases. (1) If the attacker can tell successful decrypts from the recovered information (verifiable plaintext), or (2) from some more general evidence given by the context whether he has guessed the right key (verifiable text).

kept private is often random, for example the DSA X component, or random looking, for example the RSA private exponent. Indeed, if it were not essentially random, it could be guessed, and hence could not be truly private. This is a natural situation in which to apply the form of camouflage we described above.

2 Camouflaging Private Keys

Camouflaging a private key is not entirely simple. An attacker has a number of means of distinguishing the true private key from a false one, and we have to eliminate all these means. In this section, we will discuss the necessary measures in general terms. We will find that applying these measures imposes some restrictions on the use of a camouflaged private key. We will discuss those restrictions in the next section and discuss further details of camouflaging different private keys in the sections following that one.

Let us suppose that we protect a private key by encrypting it, or parts of it, with a PIN of reasonable length, say six to eight digits. We do our actual encryption using DES or some other well-known encryption algorithm and a key derived from the PIN by some method along the general lines of PKCS #5, i.e., by hashing the PIN, possibly together with a salt.

We don't require an enormous PIN space because our security doesn't lie in making a search infeasible, but rather in making sure that many false PINs produce a plausible decrypt, and in limiting the number of PINs that an attacker can try. To be successful, we must make sure that an attacker does not have any feasible test that can distinguish the true private key from a false candidate, or at least from most false candidates. If he did have such a test, he could decrypt the private key with every possible PIN and apply the test that to every candidate private key that results. By such an attack, he would obtain the private key or at least eliminate many of the false candidates. Therefore we must make sure he has no test for the true private key, not even a probabilistic one.

2.1 Don't encrypt known structure with the PIN

Don't encrypt the whole representation of the private key in any standard form with the PIN, because they all contain some structure defined by the format. Decrypting with the wrong PIN won't usually produce a result with this structure. Don't even think of encrypting the modulus of an RSA key because, unlike the true modulus, most false decrypts will have small factors.

2.2 Conceal the public key and don't use it to

encrypt verifiable plaintext

If an attacker knows the public key, then he can tell whether a candidate private key is the real one by signing something with the candidate key and trying to verify the signature with the public key.

To avoid this attack, we have to keep the public key secret, even from its owner. Only authorized, trusted parties at secure sites may have access to a user's public key. This restriction has a number of consequences, discussed in the next section.

Likewise, if the public key is used to encrypt any verifiable plaintext, we can test for the true private key by decrypting the ciphertext and verifying the resulting text.

2.3 Don't reveal information about the PIN

This is an obvious point, but most key containers contain a hash of the password so as to check that it is (probably) correct before using it to decrypt the key. That is just the sort of thing we want to avoid.

We can avoid this attack just by not keeping a hash of the PIN with the encrypted private key. A hash does serve a purpose, though—in the case of the user mis-typing the PIN, the hash keeps the key owner's correspondent from getting messages signed with an incorrectly decrypted private key. We can serve this purpose and still protect against this attack by storing a partial hash or checksum that can detect typographical errors in the PIN. For example, we could imitate ISBN and use a p-valued checksum, where p is a prime larger than the number of digits in the pin and larger than the cardinality of the character set used in the PIN. Such a checksum detects any typo that is a simple transposition or an error in one character. It should be borne in mind, however, that keeping an N-valued checksum will, on average, reduce the number of possible PINs, and hence the security of the private key, by a factor of N.

2.4 Randomize and protect signatures

Suppose that we use a camouflaged RSA private key to sign some data in a standard way, for example according to PKCS #1. That is, we hash the data with a secure hash algorithm, pad it deterministically to the length of the private key's modulus, and then encrypt it with the private key. If an attacker can get hold of the key container, the data signed, and the signature, then he can find the private key by trying every PIN and checking whether the corresponding candidate private key produces the same signature.

To avoid this attack we must pad the hash randomly instead of deterministically, so that the signing the same data again will not produce the same signature. It is crucial to

use a secure source of randomness (see [4]) so that the attacker cannot guess the random pad.

In signature systems such as DSA, the public key can be computed from the private key. Hence, even though the signature contains a random element, an attacker can test a candidate private key by computing its public key and trying to verify the signature with that public key. To avoid this attack, we must encrypt any signature so that only the intended recipient can read it.

3 Using Camouflaged Private Keys

As we saw in the previous section, in order to camouflage a private key successfully we must address a number of things besides the private key. These issues will require measures that will impose some restrictions on the safe uses of a camouflaged private key and its public counterpart. We now examine these restrictions in the context of the primary applications of public key cryptography.

1. **Use the private key to sign a message.** Since only trusted parties can be given access to the public key, a camouflaged private key can be used to sign messages that are to be verified by trusted parties, but cannot be used to sign messages sent to a stranger. For instance, camouflaged keys can be sign checks to be verified by a bank.

2. **Use the public key to encrypt a message to the holder of the private key.** As discussed earlier, for messages that contain verifiable plaintext, this would compromise the camouflaged private key.

3. **Distribute the public key in a certificate.** Public keys corresponding to camouflaged private keys can be placed in certificates. However, the public key must be encrypted so that only trusted entities can decrypt it. For example, the public key being certified could be encrypted with a symmetric key and the symmetric key distributed amongst the trusted entities. There are several ways of placing such encrypted public keys inside certificates conforming to standards such as X.509. There are two advantages to encrypting the public key in the certificate. Firstly, the Certificate Authority controls the use of the certificate by controlling the distribution of the decryption key. Secondly, revoking the certificate is simplified, since only those verifiers that possess the decryption key need to be alerted of the revocation.

4. **Authenticate by signing a challenge sent by a server in a standard challenge-response protocol.** Camouflaged private keys can be used for this, provided the server has the ability to recover the user's decrypted public key.

To summarize, a camouflaged private key is useful for digital signatures in a *closed* public-key infrastructure—one in which signed messages may be verified only by certain trusted entities. These entities will typically be servers that need to authenticate by verifying that a challenge has been signed by the user's private key. If a server receives enough bad signatures from a user, it locks-out the user pending administrative intervention. Since the public key corresponding to a camouflaged private key can be placed in a certificate, a server in a closed PKI does not need a list of all its users or of their public keys. It identifies users and obtains their keys from the certificate is just as in an ordinary, open PKI.

Now let us discuss in more detail just how to camouflage a private key.

4 Details of Protecting the Camouflaged Private Key

A user's camouflaged private key is protected not by the sheer size of its PIN space, but by the fact that an attacker cannot tell whether he has cracked it without contacting an authentication server. The size of the search space for a sophisticated hacker trying to crack a camouflaged wallet is about

$$K = \text{(Number of PINs)} / \text{(Number of checksum values)}$$

The probability that a hacker can crack a given wallet is

$$P = \text{(expected number of attempts to authenticate before the user is suspended)}/K$$

The numbers need to be chosen so that

$$P * \text{(value of breaking a key)}$$

is small enough that it is not worth the trouble of stealing a camouflaged card and trying to crack it.

Typically the number K will not be large enough to stop a brute force attack that has some feasible way to check whether an attempted decrypt of the camouflaged private key is correct without contacting a server. Therefore we must be careful to make sure there is no way that an attacker can distinguish the true private key from false decrypts.

How the private key is encrypted depends on what kind of key it is.

4.1 DSA Camouflaged private key

A DSA private key consists of three parameters (p, a large prime, q, a smaller prime, 160 bits with the top bit set, and g, a number mod p) and a private key x, which is any number less than q. The public key corresponding to x is $y = g^x$.

It is easy to protect a DSA private key in such a way that decrypting it with any pin produces a legitimate private key—just choose x with the top bit clear and do not encrypt the top bit.

4.2 RSA Camouflaged private key

We begin with a review of RSA keys. An RSA private key consists of a modulus n, which is a number of the form pq, where p and q are large primes, and a private exponent d, a number that is relatively prime to $(p\text{-}1)(q\text{-}1)$. Exponents that differ by a multiple of

$$\varphi(n) = (p\text{-}1)(q\text{-}1)/\gcd((p\text{-}1)(q\text{-}1))$$

are equivalent—encrypting with them produces the same results. The primes p and q together with some other numbers that can be used to speed up cryptographic operations are often included in the private key. An RSA public key corresponding to (n,d) is a pair (n,e) (with the same n) such that

$$de \equiv 1 \mod \varphi(n).$$

Usually d and e are chosen smaller than n. This restriction, however, is unnecessary, since exponents that are equivalent mod $\varphi(n)$ produce the same encryption function.

The relation between the public and private keys is perfectly symmetric, i.e. what is encrypted using one can be decrypted only using the other; and it is not feasible to obtain either key from the other.

Generating RSA keys for Camouflaging

1. Determine length (in bits) for the modulus and public exponent. The length of the modulus is chosen with respect to the security required of the system, say 1024 bits. The length of the public exponent must be large enough that it is infeasible for an attacker to enumerate all exponents of this length.

2. Choose two safe primes p and q of an appropriate size and let the modulus n be pq. Safe primes are primes of the form $p = 2p' + 1$, $q = 2q' + 1$, where p' and q' are also primes.

3. Choose the public exponent e to be within the allowed length and relatively prime to $(p\text{-}1)(q\text{-}1)$.

4. Let D be the inverse of e mod $\varphi(n)$. Choose a random k such that the private exponent $d = D + k\,\varphi(n)$ has no more than b bits.

Camouflaging the RSA private key

1. Leave the modulus n unencrypted.

2. Discard the highest-order and lowest-order bits of d since these are both 1. Let f be the remaining string of bits. Encrypt f using a key derived from the password or PIN. Use an encryption method that preserves length. For example, use DES, padding f with random bits to a multiple of the block length when en-

crypting, maintaining a record of the length of f in conjunction with the encrypted text.

Recovering the Camouflaged Private Key

1. The modulus n is unencrypted and directly available.

2. Decrypt the encrypted form of f using the key derived from the user's password or PIN. Discard the padding. Restore the highest-order and lowest-order bits to obtain d.

How camouflaging works on RSA Keys

The crux of the problem is to encrypt private exponent d in such a way that an attacker cannot distinguish a false decrypt from a true one.

One obvious characteristic of RSA private keys is that the private exponent d is odd. We thwart attacks based on this characteristic since we guarantee that the recovered exponent is always odd. Another characteristic of the private exponent is that it is prime relative to $(p\text{-}1)(q\text{-}1)$. The attacker cannot exploit the second characteristic, since finding $(p\text{-}1)(q\text{-}1)$ is equivalent to factoring the modulus, an infeasible task by assumption in the RSA system. Could an attacker conceivably find some way to test this property even though he cannot find p and q? We avoid this possibility by choosing p and q to be *safe* primes. Then any false decrypt d', being odd, will, like the true exponent d, be in a given range and a legitimate exponent for n, unless d' is divisible by either p' or q', the probability of which is miniscule.

The private exponent may also have some other characteristics depending how it is generated. For example, the usual methods of generating an RSA key (e.g. [9], p. 467) produces a private exponent d that is smaller than $(p\text{-}1)(q\text{-}1)$. It is hard to tell whether a candidate private exponent d' is smaller than $(p\text{-}1)(q\text{-}1)$, which the attacker does not know. To make things more difficult for the attacker, we pad d with a multiple of $\varphi(n)$.

There is one other property that d has and a false decrypt d' may not have. The private exponent d has a corresponding public exponent e that has a limited number of bits. One may be worried that an attacker could test this property, even though he cannot compute the public exponent e' corresponding to a candidate private exponent d'. If so, just choose a general public exponent e, i.e. let its length be the number of bits in n and pad it to the length of n when encrypting it. But in practice we are likely to prefer choosing our public exponents relatively short, making public key operations, which are usually performed on a busy server, cheaper. But making e short requires some additional care—see the discussion of Wiener's attack, below.

Encrypting the private key in the proper way does not solve all the problems involved in camouflaging it. We

must also address some hazards that arise from the context in which the private key is used.

5 Hazards of Camouflaged Keys

5.1 Brute Force Public Exponent Attack for RSA

The requirement to conceal the public key has some consequences for RSA. Since we do not try to keep the modulus secret, we must depart from the common practice of using some fixed number as the public exponent. Moreover, we must use public exponents that are long enough that an attacker cannot enumerate them all in the following simple brute force attack: For each possible public exponent, test whether it can decrypt a string encrypted by a candidate private key. With high probability, the true private key will be the only candidate whose public exponent is short.

5.2 Known signature attack for RSA

If a hacker has a message signed using a camouflaged private key and knows exactly what was encrypted to form the signature, he can try encrypting that thing with all decrypts of the private key until a result matching the signature is obtained.

There are two possible ways to avoid this attack.
1. Attach a random pad to the hash of the message before encrypting it. That way the hacker may know the message and hence the hash, but does not know what exactly was encrypted. (This practice is differs from PKCS #1, which specifies deterministic padding for private key operations.)
2. If one is worried that even a signature with a random pad may give away some information about the random exponent, encrypt the signature with another key before sending it, as discussed below for DSA.

Note that if we do choose to encrypt the signature, we still need some random element, since otherwise the attacker can test a private key against the encrypted signature of a message by signing and encrypting the same message, then comparing results. If a random pad is not used when signing, use the standard form of hybrid message encryption—choose a random symmetric key, encrypt the signature with that key using one's favorite symmetric encryption algorithm, then encrypt the symmetric key with the recipient's public key.

5.3 Known signature attack for DSA

Unlike RSA, a DSA public key can be computed from the private key. Therefore, if a hacker has anything signed by a camouflaged DSA private key, he can, in spite of the random component of DSA signatures, crack the camouflaged key container by doing the following.
1. Decrypt the camouflaged private key with each PIN that matches the checksum.
2. For each resulting candidate, compute the related public key and try to verify the signature with that public key.

Verification will succeed only for the correct private key.

To avoid this possibility we need to encrypt signatures when we send them to the authentication server. This key could be the authentication server's public key, or we could pair the signing key with an encryption key. For example,

camouflaged private key = DSA private key + Elgamal public key

camouflaged public key = DSA public key + Elgamal private key

To sign data, form the DSA signature and then encrypt that with the Elgamal public key. To verify a signature, decrypt with the Elgamal private key and then verify the DSA signature.

Pairing keys in this way gives the combined key the property that RSA keys have, that one cannot derive the public key from the private one. Note that, since DSA signing includes a random component, it is not necessary to add any further randomness.

5.4 Wiener's attack for RSA

In the camouflaged scheme, public and private exponents have changed roles in some sense.
1. The public exponent is considered to be completely secret from the typical attacker, i.e. one that does not operate an authentication server.
2. The private exponent is encrypted, but the PIN space is small enough that the attacker could try them all and examine each candidate private exponent.

Thus, we need to consider classical attacks on RSA, but with the role of the public and private exponents reversed. Wiener's attack [11] on a short private (in our case, public) exponent is of particular interest because we would like to use a rather short public key, so that the public key operations which are performed on a busy server, are efficient as possible. A hacker could however, try Wiener's test for all private key candidates. The probability is vanishingly small that more than one candidate would produce a success with Wiener's attack and extract a short exponent for the complementary key.

Wiener's attack breaks down if the public exponent is large enough—half the length of the modulus suffices—or if the private exponent is large enough (larger than $n^{3/2}$).

To make the private exponent large, add a multiple of $(p-1)(q-1)$ to it. The resulting exponent is mathematically equivalent to the usual one, though encrypting with it is more expensive.

Of course, independent of the defense against Wiener's attack, the public exponent should not be so short as to be subject to the brute force attack discussed above.

5.5 Using the same PIN for a camouflaged key and for something else

Suppose the PIN for a camouflaged key is used as the password that decrypts an object that contains verifiable plaintext, such as an ordinary key store or a log-on password. Then a hacker can find all the pins that pass the checksum test and see which one unlocks the other object. If there is one that does so then it is pretty likely to be the PIN for the camouflaged key container. Even if there is no checksum, using the same PIN means that the camouflaged key is no safer than the other container. If the character set for the camouflaged PIN (e.g. digits) is more restricted than it would normally be for the other container (e.g. printable characters) then the security of both containers is reduced.

On the whole, we cannot prevent a user from using a camouflaged key's PIN for something else, only try to discourage it. Note that smart cards are subject to the same attack in a degree comparable to that of a camouflaged key without a checksum.

5.6 Two copies of a Camouflaged key with different PINs or two Camouflaged keys with the same PIN and different salts

Encryption based on a PIN is usually done following a method similar to PKCS #5, which specifies that one hash the PIN together with an 8-byte random salt to form the actual encryption key. The point of using the salt, which is not secret, is to slow down a dictionary attack by making it impossible to precompute the encryption keys derived from the PINs. Since camouflaged key protection does not depend on slowing down a dictionary attack, it does not really need to use a salt, and the following attack shows that there is good reason why we should not. Essentially the same attack shows that if a checksum is used then we must never encrypt the same camouflaged key with different PINs. Hence we should allow the user to set the PIN once only, when the key is created, or at least before it leaves a secure area.

Suppose we have two copies of a camouflaged key, encrypted with different PINs. For each key, the hacker decrypts with each PIN that passes the checksum test for that key, getting a set of candidates for the true private key. Probably there is only one key that is in both sets.

When there is a salt, a similar attack can work even without a partial hash, even if the PINs are the same, because if the set of salts is much larger than the space of PINs, as it usually is, the probability is vanishingly small that any pair of decrypts but the true ones will match.

If two camouflaged keys are encrypted with the same PIN and different salts and a checksum is kept then a comparatively small proportion of the PINs (1/(number of checksum values)^2) will pass both checksum tests, helping the attacker guess the right one.

Without salts, it is only mildly bad to use the same PIN for different camouflaged keys. The main consequence is that an attacker has more servers on which to try a limited number of passwords each.

6 Comparison with Other Authentication Methods

The camouflage technique gives us a tamper-resistant software key container, a "software smart card," that can be used mainly for authentication. How does such a software key container compare for strength and flexibility with other common authentication methods, such as passwords, public key authentication with the key in a conventional password-protected key container, and password-protected smart cards?

6.1 Advantages of Software Smart Cards

1) There are several varieties of password-only authentication methods, ranging through old fashioned password systems with a hashed or encrypted form of the password stored on the server [6], possibly with the password transmission protected by SSL, through Kerberos [10] to other, more secure, secure password protocols in which the password is never transmitted [1,3,12]. Software smart cards have the following advantages over all forms of password-only protocols.
    a) Software smart cards are a form of two-factor authentication, so an attacker cannot get access to an account simply by watching a user type his password or PIN.
    b) Software smart cards use certificates, so that the server need not know its users in advance.
2) Software smart cards are stronger than conventional software key containers because they are not susceptible to brute force or dictionary attacks. Conventional software key containers default to one-factor authentication since having access to the key-container typically implies that the password protecting the container can be guessed by dictionary attack. Any password that the user can remember without writing down is in principle subject to some form of dictionary attack.

3) Brute force and dictionary attacks get easier as computers get faster. Computer speed does not affect the security of the camouflage until it gets fast enough to factor the modulus of an RSA key or crack the encryption protecting a public key.

4) Software smart cards do not require the purchase, distribution, or administration of cards and card-reader hardware.

5) Software smart cards can be copied and backed-up as a convenience to the user.

6) Software smart cards can be stored on any storage medium, including hardware smart cards.

6.2 Disadvantages of Software Smart Cards

1. With a software smart card, all cryptography must be performed on a host processor and is therefore subject to unauthorized copying by viruses. Storage smart cards also suffer this risk. Crypto smart cards are immune to such attacks since all processing is done within the card. But even with a crypto smart card, a virus can switch the documents being signed, causing the user to sign false documents unwittingly.

2. The strength of software smart cards applies only to closed PKIs.

7   Camouflaged Symmetric Keys

A number of commentators have suggested a modification of our method in which the public/private key pair is replaced by a symmetric key. In such case, things don't change much superficially but there is a subtle but important difference in the trust models. Specifically, the symmetric key variation of the scheme is weaker than the asymmetric key version, because it allows rogue server operators to steal the secret key of any user who sends his certificate to that server as part of the authentication protocol. In cases where this risk is not considered significant, the symmetric key version is an attractive one because it is more efficient.

8   Conclusion

We have presented a camouflage technique for software storage of digital signature credentials. The technique enables "software smart cards"—software containers for secure storage of private keys, combining the assurance of hardware with the flexibility of software. Typical applications for the technology include intranet applications such as HR and enterprise resource planning; extranet applications such as supply chain management, business-to-business e-commerce, healthcare information systems; consumer applications such as e-payment, e-

commerce, and online banking; enterprise security applications such as firewall remote access etc.

9   Acknowledgements

10   References

[1] S.M. Bellovin and M. Merritt. Encrypted key exchange: "Password-based protocols secure against dictionary attacks", *Proceedings of the 1992 IEEE Computer Society Conference on Research in Security and Privacy*, 72-84, 1992.

[2] L. Gong, T.M.A. Lomas, R.M. Needham, and J.H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648-656, June 1993.

[3] D. Jablon. Strong password-only authenticated key exchange. *Computer Communication Review*, 26(5):5-26, October 1996.

[4] J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Cryptanalytic attacks on pseudorandom number generators", *Fast Software Encryption, Fifth International Proceedings*, pp. 168-188, Springer-Verlag, 1988.

[5] T.M.A. Lomas, L. Gong, J.H. Saltzer, and R.M. Needham, "Reducing Risks from Poorly Chosen Keys". *Proceedings of the 12th ACM Symposium on Operating System Principles*, pp. 14-18, 1989.

[6] R.H. Morris and K. Thompson. Unix password security. *Communications of the ACM*, 22(11):594, November 1979.

[7] PKCS #1: RSA Encryption Standard, RSA Laboratories Technical Note, Version 1.5, Nov. 1, 1993, http://www.rsa.com/rsalabs/pubs/PKCS/

[8] PKCS #5: Password-Based Encryption Standard, RSA Laboratories Technical Note, Version 1.5, Nov. 1, 1993, http://www.rsa.com/rsalabs/pubs/PKCS/

[9] Bruce Schneier, *Applied Cryptography*, second edition, Wiley, 1996.

[10] J.G. Steiner, C. Neuman, and J.I. Schiller, "Kerberos: An Authentication Service for Open Network Systems, in Proceedings of the USENIX Winter Conference, February, 1988, pp.191-202.

[11] Michael J. Wiener, Cryptanalysis of of short RSA secret exponents, *IEEE Transactions on Information Theory*, v. 36 no. 3, May 1990.

[12] Thomas Wu, "The secure remote password protocol," in *Proceedings of the 1998 Network and Distributed System Security Symposium*, Internet Society, pp. 97-111, 1998. http://srp.stanford.edu/srp/ndss.html
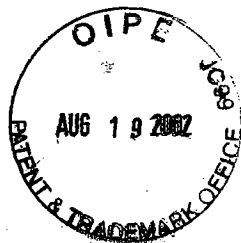
Docket No.: 028410-0016
Serial No.: 09/874,795
Mailed: August 15, 2002   Due: n/a
The return of this stamped self-addressed postcard
will acknowledge receipt in the U.S. Patent and
Trademark Office of the following documents:

1. Transmittal of Information Disclosure Statement
   (2 pp.; original & 1 copy)
2. 1449 - 1 sheet
3. Seven references

SASM&F

AUG 2 6 2002

RECEIVED